Mongrel: Hybrid Techniques for Standard Cell Placement*

Sung-Woo Hur and John Lillis
EECS Department, University of Illinois at Chicago
{shur, jlillis}@eecs.uic.edu

ABSTRACT

We give an overview of a standard-cell placer *Mongrel*. The prototype tool adopts a *middle-down* methodology in which a grid is imposed over the layout area and cells are assigned to bins forming a *global placement*. The optimization technique applied in this phase is based on the *Relaxation-Based Local Search* (RBLS) framework in which a combinatorial search mechanism is driven by an analytical engine. This enables a more global view of the problem and results in complex modifications of the placement in a single search "move". Details of this approach including a novel placement legalization procedure are presented. When a global placement has converged, a detailed placement is formed and further optimized by the proposed *optimal interleaving* technique. Experimental results are presented and are quite promising, demonstrating that there is significant room for improvement in state of the art placement.

1 Introduction

The cell placement problem is among the most fundamental in VLSI physical design. Because of the impact of placement on such design objectives as area, routability and timing, it has received extensive attention during the past two decades. In the context of standard cells, the classical wire-length driven formulation can be stated as follows: given a netlist of standard cells of fixed height (but variable width) and a specified number of rows, assign each component to a row and to an x-position in that row such that no two cells overlap and that estimated wire length is minimized. While there are typically additional details which must be considered (e.g., wire-length estimator, constraints on white space, determination of the number of rows, etc.), the preceding formulation captures the computational nature of the problem. With respect to wire-length estimation, the most common estimator for the length of a net is the *Half-Perimeter* (HP) of the pins. The popularity of HP results from its simplicity, the fact that it is an exact estimator of the minimum rectilinear Steiner tree for 2 and 3-pin nets, and mounting evidence of its relation to routing congestion [1] (this is not to say that more detailed optimizations before routing are not useful, merely that the prevailing wisdom is that a good HP minimizer is a very useful, if not essential, first step).

Placement techniques which have received substantial attention in the literature and in industry include *recursive bisection* (e.g., [2]), *analytical placement* (e.g., [3]), and *annealing-based methods* (e.g., [4]). The first two approaches can typically be considered *top-down* while annealing-based tools typically have a "flatter" view of the placement. All three classes of approaches have their merits.

This paper proposes a set of hybrid techniques for standard cell placement. The flow of our prototype tool Mongrel is best described as *middle-down*. An initial *global placement* phase focuses on assigning cells to global bins in a grid imposed over the layout area. This grid is comparatively fine-grained with typically a couple dozen cells per bin and the number of rows in the grid equal to the number in the placement. Our approach has two levels of hierarchy: after the global placement is established, a *detailed* placement phase begins in which the exact cell locations are determined. At least one previous paper ([5]) has adopted a middle-down approach (though typically starting with a coarser grid). The novelty of Mongrel lies not only in its middle-down methodology, but also in the optimization techniques applied in global and detailed placement. A brief overview of these techniques is as follows.

- In the global placement phase, we adopt the Relaxation Based Local Search (RBLS) methodology proposed in [6]. RBLS utilizes an analytical engine to drive a local search mechanism. One iteration can be roughly summarized as follows: Given a current global placement, a subset of the cells are selected and designated as *mobile* cells; using an analytical engine, the optimal relaxed placement (i.e., ignoring bin capacities) of the mobile cells is found (i.e., a linear program is solved); the information yielded by the relaxed placement is then used to induce a new legal placement via a legalization procedure. The best legal solution seen during the legalization phase becomes the next configuration and is accepted if it improves over the initial placement.

  The result is a powerful local search mechanism in which individual moves may result in complex modifications to the placement. For the approach to succeed, the analytical engine must be fast and the legalization procedure must be carefully designed. The former is achieved by adopting the network flow techniques of [6]. For the latter we have developed a novel scheme presented in Section 3.4.

- While the RBLS technique is quite effective, there is likely room for further improvement in global placement via algorithms which do not relax bin capacity constraints. Such an approach is to simply apply the Fiduccia-Mattheyses (FM) partitioner [7] to adjacent bins until there is no further improvement. This provides a complementary optimization: The view is more local than that of relaxation, but it also has greater control over constraint satisfaction.

- In detailed placement, we are allowed to perturb the ordering of cells within each row. In this phase we propose a new technique we have called *optimal interleaving* and also incorporated the *dynamic clustering* technique of [6].

Together these techniques form our prototype placer Mongrel which we suggest represents a significant departure from traditional techniques. Preliminary results are very promising: on standard MCNC benchmarks we have obtained substantial improvements (approximately $16\% - 17\%$ on average) over recently published results.

In the remainder of this paper we give some notational conventions, present algorithmic details of Mongrel's various phases and present experimental results.

2 Preliminaries

We model a netlist by a hypergraph $G = (V, E)$, where V is a set of cells and E is a set of nets. A hyperedge $e \in E$ is a subset of 2 or more cells in V (i.e., $e \subset V$). Each cell corresponds to a circuit component and each net represents a common signal among its constituent cells. Let $|e_i|$ denote the number of cells associated with net e_i and $s(v)$ the size of cell v. Also, let A denote the total cell area – i.e., $A = \sum_{v \in V} s(v)$.

In global placement, the core region is divided into an $R \times C$ grid. Given a grid and maximum and minimum allowable row size,

*This work was partially supported by NSF Grant CCR-9875945, the DAC Scholarship Program and an equipment donation from the Intel Corporation.

0-7803-6445-7/00/$10.00 © 2000 IEEE

the global placement problem is to assign cells to global bins to optimize the total wire length, subject to the following constraints:

$$(1-\varepsilon)\cdot\frac{A}{R\times C}\le\sum_{v\in b_{i,j}}s(v)\le(1+\varepsilon)\cdot\frac{A}{R\times C}\qquad(1)$$

$$\text{lower bound}\le\sum_{1\le j\le m}\sum_{v\in b_{i,j}}s(v)\le\text{upper bound}.\qquad(2)$$

The *bin-capacity constraint* (1) ensures that all bins are close to their target capacity while allowing flexibility for optimization via the parameter ε. The *row-size constraint* (2), which can be viewed as a white space constraint, ensures that all rows are of rougly equal size.

A global placement P is *legal* if it satisfies the bin-capacity and the row-size constraints. A detailed placement P is *legal* if no two cells overlap and the row-size constraint is satisfied.

We use half perimeter of a bounding box of a net as length of the net. The length of net i, $len(e_i)$, is estimated by

$$len(e_i)=\max_{u,v\in e_i}|x_u-x_v|+\max_{u,v\in e_i}|y_u-y_v|$$

where (x_v,y_v) gives the coordinates of cell v (in the case of a global placement a cells coordinates are the center of the bin to which it belongs). The placement problem can then be stated as

$$min\sum_{i=1}^{|E|}len(e_i)\text{ over all legal placement }P$$

where the notion of legality depends on whether the placement is global or detailed.

3 Relaxation Based Local Search

3.1 Overview

The notion of *relaxation based local search* (RBLS) was introduced in [6]. In that paper it was applied to the linear placement problem. In Mongrel, we adopt a RBLS approach for row-based placement. This section describes the issues involved.

At the top-level of abstraction, RBLS adopts a quite traditional local search framework. From a current solution we sample a neighboring solution and move to that solution if the objective is improved. If no improvement is seen for k (a given parameter) consecutive moves, the search terminates.

The novelty of RBLS is in how neighboring solutions are generated. Rather than employing simple "moves" such as cell swapping, we use an analytical engine to make more drastic modifications to the placement in the hope of capturing a more global view. This process is summarized as follows:

- **Sub-circuit Extraction**: Given a parameter m, extract a sub-circuit $M(\subseteq V)$ where $|M|=m$. M will be called the set of "mobile nodes". From M we determine the *fixed* node set F: the set of nodes in $(V-M)$ which are directly connected to a member of M via some net. An extracted sub-circuit consists of node set $F\cup M$ and net set E' induced by M.
- **Optimal Relaxed Placement**: Given a set of mobile nodes M, we find the optimal placement for each member of M ignoring capacity constraints under a linear programming relaxation of the problem. Note that the relative order of cells in F influences this solution.
- **Placement Legalization**: The information yielded by the relaxed placement is then used to induce a new legal placement via a legalization procedure.
- **Further Optimization**: A legal global placement is then further optimized using partitioning technique between neighboring bins.

This flow is summarized by the pseudo-code in Figure 1.

Algorithm *Relaxation Based Local Search (RBLS)*
Input: m,k, and current placement P
Output: new placement

```
Counter ← k
while (Counter > 0) {
    Extract a mobile node set M (|M| = m)
    Determine a fixed node set F using M
    For ∀v ∈ M, determine optimal relaxed location
    P' ← new placement after resolving size problem
    P' ← optimize P' by local partitioning
    if (WL(P') < WL(P)) then
        Counter ← k
        P ← P'
    else
        Counter ← Counter − 1
}
return P
```

Figure 1: Algorithm *Relaxation Based Local Search (RBLS)*

3.2 Sub-circuit Extraction

In general, the sub-circuit extraction strategy (determination of mobile nodes in RBLS) affects the final solution quality. Experiments with several candidate strategies led us to the simple and effective randomized scheme described below.

First a net e with degree no greater than m is selected at random and all members of e are added to M. We then repeatedly select (at random) nets adjacent to M, but previously unselected and add new members to M until M reaches its target size m. Thus the approach can be considered a randomized net-based "wavefront" algorithm. The result is a connected component in the hypergraph which will then be handed to the analytical solver for optimal relaxed placement. Experiments have shown this technique to perform better than a node-based wavefront approach (as in [6]).

3.3 Optimal Relaxed Placement

Once M is extracted, we decompose the relaxed placement problem into independent x and y subproblems. As in [6], the mobile node set M induces a set of *fixed nodes* F and a set of *active nets* E' (all nets which influence the relaxed placement problem for M) – i.e., $E'=\{e_i\mid e\cap M\neq\emptyset\}$ and $F=\{v\in(V-M)\mid\exists e\in E'\text{ s.t. }v\in e\}$. The effect of the fixed nodes is to provide "anchors" preventing the mobile cells from collapsing to a single point. Note that in practice it is only necessary to find left-most/right-most (top-most/bottom-most) fixed nodes when solving for the x (y) dimensions). Note also that fixed nodes may either be pad cells or regular cells.

To determine the optimal relaxed $x-$ and $y-$coordinates for mobile nodes, we first project the extracted sub-circuit onto the $x-$ and $y-$axes as shown in Figure 2. In the example, assume for simplicity each cell has unit size and only one cell can be placed in a bin. Cells b,c and d are selected as mobile nodes. Nets e_1-e_7 are active since they are connect to mobile cells. Cell a a fixed for active net e_1. Similarly, cells a and f are fixed for e_3, and so on.

As in the preceding discussion, to find the location for each mobile node to minimize the total half-perimeter wire length of the active nets (ignoring capacity constraints), we decompose the problem into two independent linear programs (LP). Note that these two LPs differ as a result of the different ordering of fixed nodes in the x and y dimensions. The LP relaxation for the x-dimension can be stated as follows.

$$\min\sum_{e\in E'}(r_e-l_e)\quad\text{subject to}$$
$$l_e\le x_v\le r_e,\qquad\forall v\in e,$$
$$x_v=X_v,\qquad\forall v\in F,$$

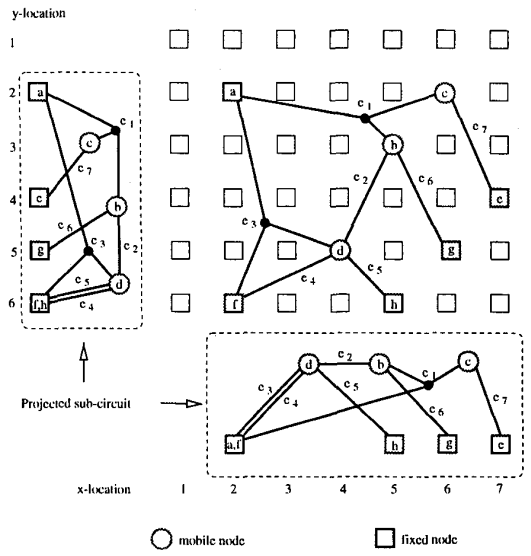


Figure 2: Sub-circuits projected onto x and y axes.

X_v being the location of a node $v \in F$ in a given feasible placement P. The dummy variables r_e and l_e give the leftmost and rightmost ends of net e. The y-coordinates for mobile nodes can be derived from an analogous LP.

In [6] it was shown that this LP can be solved efficiently via network flow methods. As a result, it was shown to be practical to solve such problems in the inner loops of a local search scheme (while the use of a general LP solver would likely not have been practical). We adopt these network flow techniques in Mongrel.

3.4 Legalization

A crucial issue in relaxation-based methods is the resolution of cell overlaps or bin/row constraint violations. We have developed two new techniques for this legalization process which have proved essential for the effectiveness of our placer.

First, in conventional legalization procedures, one begins with the relaxed placement and gradually massages it into a placement obeying the various constraints. In such a procedure a legal placement is derived only at the very last step. While this might be sensible for top-down analytical methods, it seems less than ideal for our search-based procedure. In our legalization procedure, we start with the initial legal placement P (before relaxation) and then sequentially move each mobile node to its relaxed target location; if this results in a constraint violation we then invoke a constraint resolution procedure. The key point is that after each cell is moved we produce a new *legal* placement. Thus during the course of legalization we have m intermediate legal placements; adopting a philosophy similar to that of a single pass of a partitioner like Fiduccia-Mattheyses we then select the minimum wire-length intermediate solution. This vastly increases the likelihood of producing an improving move in our search scheme. This procedure is summarized as follows.

- Store the wire length of the current legal placement and regard it as best.
- For each mobile node v in the same sequence as added to M in the sub-circuit extraction phase:
  (1) Place v at its optimal relaxed location (x_v, y_v) giving intermediate relaxed placement P'.
  (2) If P' is legal, goto step (3). Otherwise, invoke the correction procedure (see below) and obtain a new legal placement.
  (3) If the new legal placement has a shorter wire length, record it as the best.

Our second contribution in legalization is how to handle step (2) above. When a cell is moved to its target bin, it may produce a constraint violation (under/over flow of a bin or row) and some kind of correction must be made. Intuitively, the modifications made to legalize the placement should attempt to disturb cell locations as little as possible while minimizing wire length. Toward these ends we propose a "node rippling" procedure summarized as follows.

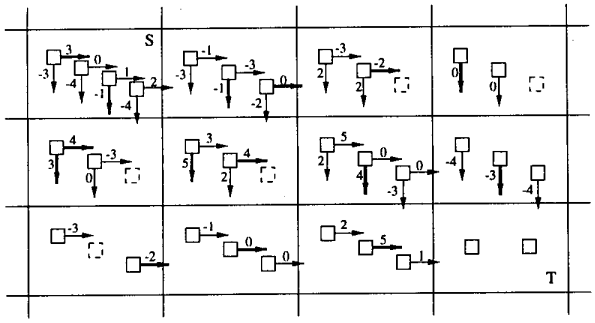
1. if there is an excess bin, say S, find a nearest bin, T, to which a cell can move in without violating T's bin-capacity constraints. Ripple move cells from S to T along a monotone path (sequence of bins).
2. if there is a deficit bin, say T, find a nearest bin, S, from which a cell can move out without violating the bin-capacity constraints. Ripple move cells from S to T along a monotone path.
3. if row-size constraints are not satisfied, find a max-bin in the largest row, say S and find a min-bin in the smallest row, say T. Ripple move cells from S to T along a monotone path.
4. Repeat above procedure until the size constraints are satisfied.

Once we have selected a source bin S and a destination bin T, a ripple move is then a sequence of cell moves from S to T where we always move a cell from the current bin to a bin in the direction of T – i.e., the bin sequence is monotone. Once the above sequence of ripple moves is complete we have a new legal placement.

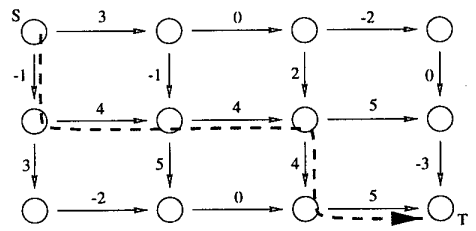
To determine the bin sequence and the cells to be moved along that sequence, we perform a global analysis based on the *gains* of individual cells in the bins. This notion is illustrated in Figure 3. In the figure since S is in the upper left corner and T is in the lower right, all candidate moves are either to the right or down. Candidate cells to be moved are indicated by solid boxes and are either nodes not in M or nodes in M but previously placed during the legalization procedure. (Dashed boxes are used to represent as-yet unplaced mobile nodes which are not candidates for rippling). The gain associated with a particular move is simply the resulting wire-length reduction which may be positive or negative. If we assume that an individual cell is moved at most once, we then have independence among the gain values; this can be seen by noticing that any montone path crosses each cut-line in the region exactly once. This scenario induces the *gain-graph* in part (B) of the figure where each vertex corresponds to a bin and an weighted arc represents the maximum gain in that direction (horizontal and vertical gains may be maximized with different cells). Further, since the gain-graph is acyclic, we can find the maximum gain path by topological ordering. Once the bin sequence is determined in this way we actually perform the ripple move with the slight modification that we allow a cell to move more than once (the result is that the total gain is at least that of the global analysis, and some times better).

Calculating Gain Values

An important issue is the relationship between the gain values and the positions of the mobile nodes which have not yet been moved to their target positions in the sequence. The location of such cells will affect the gain of cells in the ripple sequence. We have chosen to be optimistic and calculate these gains based on the optimal **relaxed location** of the as-yet unplaced mobile nodes (*previously* placed mobile nodes are naturally assumed to be in the location where they end up). Thus the the gain values may be inexact, but this approach is clearly preferable to calculating gains based on the *original* positions of as-yet unplaced mobile nodes.



(A) gain values to determine a monotone path S to T



(B) transformed graph

Figure 3: (A) Gain values to determine a monotone path. Dashed rectangles denote mobile nodes which are excluded from candidates to move (B) Resulting gain-graph and a max-gain monotone path (total gain ≥ 16).

3.5 Placement Optimization By Local Partitioning

Reducing wire length in a global placement is closely related to minimizing the cut size between adjacent bins. Thus it is natural to perform such re-partitioning between neighboring bins to complement the relaxation based component of the global placer. Even though the view is more local than that of relaxation, it has greater control over constraint satisfaction. Based on this observation, we adopt a generic Fiduccia-Mattheyses (FM) algorithm [7] over a pair of adjacent bins – i.e., to cells in two bins $b_{i,j}$ and $b_{i,j+1}$ (or $b_{i,j}$ and $b_{i+1,j}$) aiming to minimize cut size.

Scanning the global placement from the top row to the bottom, from the left to the right in each row, assuming two adjacent bins as an initial partition, we apply the generic FM-partitioner to the pair of bins subject to the bin-capacity and row-size constraints.

This optimization step can be repeated as long as there is improvement greater than a threshold value b for each scan.

4 Detailed Placement

Mongrel has two levels of hierarchy: *global placement* and *detailed placement*. In this section, we describe a method to transform a global placement to a detailed placement and propose a novel optimization technique for a linear arrangement which is adopted to optimize every row in the detailed placement.

Note that the by construction of the grid, global placement completely determines the row assignments for the detailed placement. Thus, if a global placement satisfies the row-size constraint, the corresponding detailed placement will satisfy the constraint.

4.1 Converting Global to Detailed Placement

A global placement is first transformed to a detailed placement. When we place each cell in a row in a detailed placement, we assume all cells are abutted. The relative order of cells in the global placement is preserved in the initial detailed placement – e.g., cells in a left bin in a global placement are placed left. To determine the initial ordering among cells in the same bin we use a simple greedy based scheme based on a force-value exerted on each cell.

4.2 Optimal Interleaving

Once an initial detailed placement is obtained we perform further intra-row optimization via a technique we call *optimal interleaving*. Figure 4 illustrates the key steps of the interleaving technique which can be summarized as follows.

- Given window size W, find a subsequence A within W from current linear arrangement of cells in a row. The relative order of cells in A is preserved.
- Let $B = \overline{A}$ in W preserving relative order of cells in the original sequence.
- Interleave sequences A and B to get an optimal arrangement.
- The above steps are repeated by sliding the window right across each row from the top row to the bottom.

The figure illustrates the solution space covered by interleaving for given A and B. Thus there is potential for quite non-trivial optimization: the number of all possible ways of interleaving is $\binom{n+m}{n}$, where $|A| = n$ and $|B| = m$. A polynomial time algorithm for finding the optimal such interleaving among this exponentially large set is sketched in the sequel.

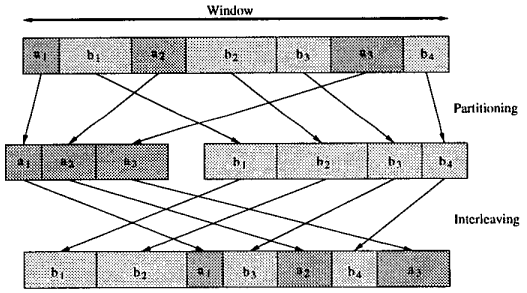


Figure 4: Illustration of interleaving technique

Dynamic Programming for Interleaving

Optimally interleaving two subsequences can be done efficiently via dynamic programming.

Given window W size of $n+m$, suppose $A = a_1, a_2, \cdots, a_n$ and $B = b_1, b_2, \cdots, b_m$. Let $S_{i,j}$ denote an optimal arrangement with $a_1, a_2, \cdots, a_i (i \leq n)$ and $b_1, b_2, \cdots, a_j (j \leq m)$ and $C(S_{i,j})$ the cost of $S_{i,j}$. The objective of interleaving A and B is to find $S_{n,m}$.

The cost $C(S_{i,j})$ of a partial placement $S_{i,j}$ is its total (x-dimension) wire-length within the window W up to the right boundary of $S_{i,j}$. This can be viewed as the normalized sum of the wiring densities along the sub-window covered by $S_{i,j}$. A key point is that the optimal interleaving of a prefix $S_{i,j}$ is independent of the ordering of subsequent cells in the window. This separability allows a dynamic programming approach.

A recurrence relation for the dynamic programming can be stated as follows.

$$
\begin{aligned}
S_{0,0} &= \emptyset \\
C(S_{0,0}) &= 0 \\
S_{i,j} &= \begin{cases} S_{i-1,j}a_i, & \text{if } C(S_{i-1,j}a_i) < C(S_{i,j-1}b_j) \\ S_{i,j-1}b_j, & \text{otherwise.} \end{cases}
\end{aligned}
$$

$S_{n,m}$ can be obtained by using an $n \times m$ table. A relatively straightforward analysis reveals that the algorithm can be implemented in $O(nm + p(n+m))$ where p is the total number of pins on incident nets. Note that p is typically linear in the number of cells.

Optimal interleaving is iteratively applied to a detailed placement by sliding windows. We have found that selecting a window size W which is twice the average number of cells in a global bin is effective. The partitions into sets A and B are done completely at random.

4.3 Dynamic Clustering

When a detailed placement converges after iteratively applying interleaving, there is often still room for further improvement. In [6], a *dynamic clustering* method was proposed for linear placement. The technique was shown to enable escape local optima. The scheme is summarized as follows. Given parameters L and U (lower- and upper-bound for the size of a clustered node), we dissect each row into clusters of cells where each cluster has no fewer than L and no more than U cells. Thus clusters are determined from the current placement by finding the minimum local wiring density within the lower and upper bounds. A generalization of optimal interleaving which considers the possibility of reversing the ordering of cells within a cluster is then applied to the *clustered* circuit. This process alternates with optimization of the flattened circuit and reclustering until there is no significant improvement. As a final note, we have found small values of L (say less than 3) and values of U of less than 10 to be effective.

5 Overall Algorithm

Figure 5 shows the overall procedure of Mongrel. The algorithm can be summarized as follows. First, an initial global placement is generated. Then, Algorithm *RBLS* is invoked to optimize a global placement. Control variables are used for sub-circuit size, convergence criteria, and so on. When a global placement has converged, it is transformed to a detailed placement. Interleaving and dynamic clustering techniques are then used to further optimize the detailed placement.

Input parameters R and C are used to determine the grid dimension for a global placement. R determines the number of rows and is usually set to the number that are used for the number of rows in [3, 4]. C is usually set such that each bin can accommodate $10 \sim 17$ cells based on our experience because we have found best results with such a grid dimension. i_0 is used to determine the initial value for k. (maximum number of trials in *RBLS* without improvement). During the course of RBLS, the sub-circuit size m is decreased. The rate at which it is decreased is determined by β (< 1). If m becomes less than the threshold size (k_{min}) of a mobile node set, we regard the current placement to be converged. α is used to determine the initial size of mobile node set.

6 Experiments

We have implemented *Mongrel* and tested it on a standard set of benchmarks on a 400 MHz Pentium II/Linux.

The control variable α is usually set to $0.7 \sim 0.8$, i_0 to $2 \sim 3$ for the initial sub-circuit size and k. We usually set β to 0.1, γ to 2, and k_{min} to $5 \sim 10$ depending on circuit size. To control the CPU time spent by the partitioning technique (Section 3.5) we use a variable b as explained and it is set to $0.01 \times R \times C$. To satisfy row size constraint (inequality (2)), our placer automatically

Algorithm *Mongrel*
Input: $n, m, \alpha, \beta, \gamma, i_0, k_{min}, etc$
Output: an optimized detailed placement
/***** **Global Placement*******/
$P \leftarrow$ Obtain an initial placement
$m \leftarrow \alpha \cdot |V|$
$k \leftarrow i_0$
while (TRUE) {
  Call *RBLS*(P, m, k)
  $m \leftarrow \beta \cdot m$
  $k \leftarrow \gamma \cdot k$
  if $(m < k_{min})$ **break**
}
/***** **Detailed Placement*******/
$W \leftarrow 2\times$(avg. #cells in a bin)
$P_d \leftarrow$ Transform P to a detailed placement
while ($\exists$ Improvement) {
  $P_d \leftarrow$ *Interleave*(P_d, W)
  Determine L and U
  $P_c \leftarrow$ *Cluster*(P_d, L, U)
  $P_c \leftarrow$ *Row Optimize*(P_c, W)
  $P_d \leftarrow$ *Flatten*(P_c)
}
return P_d

Figure 5: Algorithm *Standard Cell Placer*

controls the lower-bound and the upper-bound of a row size such that $\frac{upper-bound}{lower-bound} < 1.03$. As a result the average white-space is 1.5%.

We have run experiments on the same set of circuits tested with TimberWolf V.7 [4], the force-directed method of [3] and the Snap-on method of [8]. Using the hybrid techniques, not only have new best-published results been found for every circuit, but also a substantial overall improvement has been achieved. Table 1 summarizes the results. Note that the Snap-on method uses different number of rows for some circuits (e.g., 88 for avql and 64 for ind3) and hence there are multiple entries for those circuits.

While the results in Table 1 are the best of 10 independent (and trivially parallelizable) runs, Table 2 gives an idea of the statistical behavior of multiple runs of the current version of Mongrel including run-time. The table includes best-of-5, average-of-10 and worst-of-10 results as well as the best-of-10 results from Table 1.[1] The data shows the stability of the approach (though efforts continue to improve the robustness of the approach). While Mongrel is fairly compute intensive, the run-times also show promise. There is also likely significant room for improvement of CPU time through both algorithmic and implementation techniques.

7 Conclusions

We have presented hybrid techniques for standard cell placement. Our prototype tool *Mongrel* adopts a two-level middle-down approach. An initial *global placement* phase focuses on assigning cells to global bins in a grid imposed over the layout area. The placement is then transformed into a detailed placement and further optimized.

For global placement, we have adopted a Relaxation-Based Local Search (*RBLS*) mechanism [6]. By using constraint relaxation we are able to capture a global view of the problem while maintaining the flexibility and non-determinism of local search.

A proposed legalization technique improves the performance of RBLS by maintaining a sequence of legal placements from which

[1] The best-of-5 data is somewhat pessimistic in that we took the 10 trials in order and arbitrarily eliminated the half containing the best solution from consideration. A more robust statistic would randomly select subsets of size 5 from a large number of runs and average the best results. Nevertheless, the given statistics should give insight into the behavior of Mongrel.

CKT	#nets	#cells	#rows	Wire Length				(%) Improv. over		
				TW-V7	*FD*	*Snap-On*	*Mongrel*	*TW-V7*	*FD*	*Snap-On*
prim1	904	833	16	0.84	0.87	0.95	**0.83**	1.2	4.6	12.6
struct	1920	1888	21	0.364	0.338	—	**0.266**	26.92	21.30	—
prim2	3029	3014	28	3.57	3.72	3.66	**2.94**	17.65	21.0	19.7
biomed	5742	6417	46	1.62	1.78	1.84	**1.36**	16.05	23.60	26.09
ind2	13419	12142	72	13.53	14.6	14.48	**11.89**	12.12	18.56	17.89
ind3.A	21940	15059	54	42.84	45.1	—	**34.53**	19.40	23.44	—
ind3.B			64	—	—	44.70	**32.99**	—	—	26.20
avqs	22124	21854	80	5.41	4.91	5.15	**4.40**	18.67	10.39	14.56
avql.A	25384	25114	86	5.86	5.38	—	**4.87**	16.89	9.48	—
avql.B			88	—	—	5.21	**4.88**	—	—	6.33
Average								16.11	16.55	17.62

Table 1: Results compared with TW-V7 [4], Force-directed (FD) [3], Snap-on [8] methods. Mongrel results are the best among 10 runs (see Table 2 for information on solution distribution). Circuits *ind3* and *avql* have multiple versions to account for discrepancies in row-count in the literature.

CKT	Wire Length				CPU (sec)	
	Best/5	Best/10	Avg	Max	Avg	Max
prim1	0.86	0.83	0.87	0.90	162	212
struct	0.275	0.267	0.278	0.287	90	111
prim2	3.09	2.94	3.13	3.22	249	332
biomed	1.47	1.36	1.48	1.58	480	548
ind2	12.16	11.89	12.45	13.13	3443	4008
ind3.A	35.21	34.53	37.90	43.12	4814	6635
ind3.B	33.55	32.99	34.85	39.09	4738	6075
avqs	4.48	4.40	4.62	4.76	8222	8869
avql.A	4.91	4.87	5.00	5.36	8344	12448
avql.B	4.90	4.88	5.02	5.21	7636	8158

Table 2: Statistical behavior of Mongrel over multiple runs. Average and Max wire-lengths are taken over all 10 trials.

the best is selected and by incorporating a gain-based global analysis when resolving constraint violations. The net result is vastly increased likelyhood of finding an improved placement in the inner loop of RBLS. A partitioning technique to optimize a legal placement is another complementary tool in producing highly tuned solutions.

The *optimal interleaving* technique was also proposed for intra-row optimization. With a dynamic programming technique, we are able to efficiently identify the optimal interleaving of two cell sequences. This technique is then applied iteratively.

Based on an optimized global placement, a combination of the interleaving technique and the dynamic clustering technique of [6] produces best published results for every benchmark circuit. Thus, the framework of Mongrel appears to be quite powerful. We continue to pursue refinements and generalizations (both in terms of problem objectives and algorithmic tools) of the approach.

Acknowledgements

The authors wish to thank Milos Hrkic and Karthik Kalpat of UIC for their help in debugging and optimizing the code.

References

[1] M. Wang and M. Sarrafzadeh, "Behavior of Congestion Minimization During Placement," in *Proc. of Intl. Symposium on Physical Design*, pp. 145–150, 1999.

[2] M. A. Breuer, "Min-cut Placement," *Design Automation and Fault-Tolerant Comuting*, pp. 343–382, October 1977.

[3] H. Eisenmann and F. M. Johannes, "Generic Global Placement and Floorplanning," in *Proc. ACM/IEEE Design Automation Conference*, pp. 269–274, 1998.

[4] W.-J. Sun and C. Sechen, "Efficient and Effective Placement for Very Large Circuits," *IEEE Transactions on Computer-Aided Design*, pp. 349–359, 1995.

[5] M. Sarrafzadeh and M. Wang, "NRG: Global and Detailed Placement," in *Proc. of IEEE Intl. Conference on Coumputer-Aided Design*, pp. 532–537, IEEE Computer Society Press, 1997.

[6] S.-W. Hur and J. Lillis, "Relaxation and Clustering in a Local Search Framework: Application to Linear Placement," in *Proc. of ACM/IEEE Design Automation Conference*, pp. 360–366, 1999.

[7] C. M. Fiduccia and R. M. Mattheyses, "A Linear Time Heuristic for Improving Network Partitions," in *Proc. of ACM/IEEE Design Automation Conference*, pp. 175–181, 1982.

[8] X. Yang, M. Wang, K. Egur, and M. Sarrafzadeh, "A Snap-on Placement Tool," in *Proc. of Intl. Symposium on Physical Design*, pp. 153–158, 2000.